

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-219810

(43)Date of publication of application : 18.08.1995

(51)Int.Cl. G06F 11/28
G06F 9/06

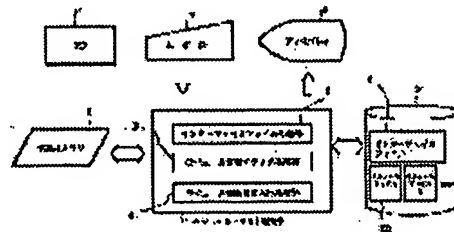
(21)Application number : 06-012701 (71)Applicant : CANON INC
(22)Date of filing : 04.02.1994 (72)Inventor : OTA SATOSHI

(54) MODULE TESTING METHOD AND DEVICE THEREFOR

(57)Abstract:

PURPOSE: To improve efficiency of the single program module test.

CONSTITUTION: An interface file processing part 2 reads a file that includes the function to be tested and procudes the name of this function and a list of input/output parameters. A module definition editor processing part 3 inputs the name of the function and the parameter name of the function based on the parameter list. A module function automatic execution processing part 4 displays the input parameter name, etc., and inputs the proper value. Then the value is outputted in accordance with the set parameter value when the execution of the function is instructed. Therefore it is not required to produce the test programs for each module to be tested, and the test efficiency is improved.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-219810

(43) 公開日 平成7年(1995)8月18日

(51) Int.Cl. ⁹	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 11/28	3 4 0 A	9290-5B		
9/06	5 4 0 U	9367-5B		

審査請求 未請求 請求項の数6 O L (全 12 頁)

(21) 出願番号 特願平6-12701

(22) 出願日 平成6年(1994)2月4日

(71) 出願人 000001007

キヤノン株式会社

東京都大田区下丸子3丁目30番2号

(72) 発明者 太田 聡

東京都大田区下丸子3丁目30番2号 キヤ
ノン株式会社内

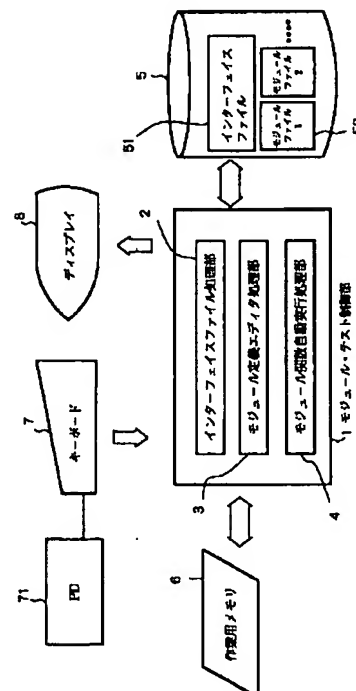
(74) 代理人 弁理士 大塚 康徳 (外1名)

(54) 【発明の名称】 モジュールテスト方法および装置

(57) 【要約】

【目的】 プログラムモジュール単体テストの効率を向上させる。

【構成】 インターフェイスファイル処理部2は、テスト対象の関数が存在するファイルを読み込んで、テスト対象の関数名や入出力パラメータのリストを作成する。モジュール定義エディタ処理部3は、前記リストに従ってテスト対象の関数名や、その関数のパラメータ名を入力させる。モジュール関数自動実行処理部4は、入力されたパラメータ名等を表示し、適当な値を入力させる。その後、関数の実行が指示されると、設定されたパラメータの値に応じた値を出力する。このようにして、テスト対象モジュール毎にテストプログラムを作成する必要がなくなり、テストの効率を向上できる。



【特許請求の範囲】

【請求項 1】 ディスプレイを有する情報処理システムにおけるプログラムモジュールテスト方法であって、テスト対象プログラムに記載されたインターフェイス情報を読み込む読み込み工程と、前記読み込み工程により読み込まれたインターフェイス情報に基づいてデータを設定させる画面を前記ディスプレイに表示する表示工程と、該表示工程により表示される画面に応じてデータを入力する入力工程と、該入力工程により入力されたデータと前記インターフェイス情報とに基づいて前記テスト対象プログラムを実行する工程と、を備えることを特徴とするモジュールテスト方法。

【請求項 2】 前記表示工程は、設定させるデータをメニュー形式で表示し、前記入力工程は前記メニューに対応したデータを入力することを特徴とする請求項 1 記載のモジュールテスト方法。

【請求項 3】 前記インターフェイス情報は、前記テスト対象プログラムの名称と該プログラムの入出力パラメータとを含むことを特徴とする請求項 1 または請求項 2 記載のモジュールテスト方法。

【請求項 4】 プログラムをモジュール毎にテストするモジュールテスト装置であって、テスト対象プログラムに記載されたインターフェイス情報を読み込む読み込み手段と、前記読み込み手段により読み込まれたインターフェイス情報に基づいてデータを設定させる画面を前記ディスプレイに表示する表示手段と、該表示手段により表示される画面に応じてデータを入力する入力手段と、該入力手段により入力されたデータと前記インターフェイス情報とに基づいて前記テスト対象プログラムを実行する手段と、を備えることを特徴とするモジュールテスト装置。

【請求項 5】 前記表示手段は、設定させるデータをメニュー形式で表示し、前記入力手段は前記メニューに対応したデータを入力することを特徴とする請求項 4 記載のモジュールテスト装置。

【請求項 6】 前記インターフェイス情報は、前記テスト対象プログラムの名称と該プログラムの入出力パラメータとを含むことを特徴とする請求項 4 または 5 記載のモジュールテスト装置。

【発明の詳細な説明】**【0001】**

【産業上の利用分野】 本発明は、例えばサンプル・プログラム等を作成することなく、作成されたモジュールの動作の確認ができるモジュールテスト方法およびそれを用いた装置に関するものである。

【0002】

【従来の技術】 従来、例えばアプリケーションプログラム等、最終的な目的となるプログラム（以下、目的プログラムと呼ぶ）を作成する際には、それを複数の機能モジュールに分割して機能モジュール毎に作成し、それらをリンクするという手法がしばしば行われる。この場合、目的プログラムの動作は各機能モジュール毎に単体テストを行って検証されるのが普通である。この各機能モジュール毎の単体テストに於ては、作成者自身が目的プログラムとは別個に簡単なサンプル・プログラムを作成し、このサンプル・プログラムにテストの対象となる各機能モジュールをリンクさせ、リンク後のサンプル・プログラムを実行して機能モジュールの動作確認を行っていた。

【0003】

【発明が解決しようとする課題】 上記従来例では、目的とするプログラムとは別に“サンプル・プログラムの作成”という本来の目的とは無関係な作業が発生しており、また、目的プログラム毎にサンプル・プログラムを作成、変更しなければならず、開発効率の低下の原因となっていた。

【0004】 本発明は上記従来例に鑑みてなされたもので、統一された、再利用可能なモジュールテスト方法およびそれを用いた装置を提供することにある。

【0005】

【課題を解決するための手段】 上記目的を達成するために本発明のモジュールテスト方法は次のような構成から成る。すなわち、ディスプレイを有する情報処理システムにおけるプログラムモジュールテスト方法であって、テスト対象プログラムに記載されたインターフェイス情報を読み込む読み込み工程と、前記読み込み工程により読み込まれたインターフェイス情報に基づいてデータを設定させる画面を前記ディスプレイに表示する表示工程と、該表示工程により表示される画面に応じてデータを入力する入力工程と、該入力工程により入力されたデータと前記インターフェイス情報とに基づいて前記テスト対象プログラムを実行する工程とを備える。

【0006】 また、本発明のモジュールテスト装置は次のような構成から成る。すなわち、プログラムをモジュール毎にテストするモジュールテスト装置であって、テスト対象プログラムに記載されたインターフェイス情報を読み込む読み込み手段と、前記読み込み手段により読み込まれたインターフェイス情報に基づいてデータを設定させる画面を前記ディスプレイに表示する表示手段と、該表示手段により表示される画面に応じてデータを入力する入力手段と、該入力手段により入力されたデータと前記インターフェイス情報とに基づいて前記テスト対象プログラムを実行する手段とを備える。

【0007】

【作用】 上記構成により、テスト対象プログラムのインターフェイス情報に基づいてデータを設定させる画面を

表示し、その画面に応じたデータを入力させ、入力されたデータに基づいてテスト対象プログラムを実行する。

【0008】

【実施例】以下、本発明の実施例を図面を参照して詳細に説明する。

【0009】図1は、本実施例である、C言語で記述されたモジュールの機能を検証するテストシステムの構成を示す。C言語においては手続きは関数という形式で表される。目的プログラムは、1または複数の関数を記述したモジュールファイルとして作成されたプログラムをリンクして得られるが、本テストシステムはこのモジュールファイル毎に、更にそのなかの関数毎に単体テストを行うものである。

【0010】まず、C言語で記述されたプログラムの構造を簡単に説明しておく。

【0011】モジュールファイル内で定義された関数には関数名が付加されている。また、その関数自体の定義の前に、その関数から返される値のデータ型が、入力パラメータとともに宣言されているのが普通である。この宣言をプロトタイプ宣言と呼ぶ。プロトタイプ宣言は関数を後方参照で呼び出す場合に、その呼び出しに先立って必要とされるもので、前方参照で呼び出す場合にはプロトタイプ宣言が省かれることもある。関数を呼び出す際には、関数が返す値を受ける出力変数と関数名と入力パラメータの並びとを、通常の関数と同様の形式で記述すればよい。また、複数の関数にわたって共通に使用される変数をグローバル変数とよび、その旨定義を記述する必要がある。

【0012】図1に於て、1は本実施例のモジュール単体テスト制御部である。モジュール単体テスト制御部1は、指定されたモジュール・ファイルから、テスト対象とする関数（テスト関数と呼ぶ）のプロトタイプ宣言部分を読み込み、インターフェイス・リストを作成するインターフェイス・ファイル処理部2と、図2に示すような対話型の画面フォームの各入力・列挙フィールドにインターフェイス・リストの内容を表示し、参照されるグローバル変数等の定義・編集を行いインターフェイス・ファイルを作成するモジュール定義エディタ処理部3と、インターフェイス・ファイルに記載された各テスト関数の項目を読み込み、図3に示すような対話形式の画面を使って、入力フィールドに代入された値をもとに、テスト関数を逐次実行するモジュール関数自動実行処理部4とを有する。5は磁気ディスク等の外部記憶装置であり、そこにはインターフェイス・ファイル51や、モジュール・ファイル52が格納されている。6はモジュールテスト制御部が使用する作業用メモリ、7はキーボード、71は表示画面に対応した座標を指定するポインティングデバイス、8はディスプレイである。なお、モジュールテスト制御部1は、CPUによりプログラムを実行することで実現できる。この場合、このCPUはテ

ストシステム全体を制御するものであってもよい。

【0013】図2はモジュール定義エディタ3による表示画面の例を示す。

【0014】オペレータはこの画面によりテスト関数を選び、インターフェイス・ファイル処理部2にインターフェイス・リストを作成させ、それを元にして各フィールドを埋め、モジュール定義エディタ処理部3にインターフェイス・ファイル51を作成させる。なお、図に於いて、各枠は候補を表示する為のフィールドであり、その右横の下向き矢印はプルダウン表示を行うためのスイッチである。この矢印をオペレータが指定することで、現在フィールドに表示されていない他の候補を表示することができる。

【0015】図2に於て、9はモジュール・ファイル選択フィールドを示し、オペレータはテストの対象となるモジュールのソースファイル名を選択する。10はテスト関数名選択フィールドを示し、テストの対象となるテスト関数名を選択する。11は変数名入力フィールドを示し、ソースファイルを読み込んだ後、それに記述された変数名を表示するが、変更も可能である。12は変数型タイプ選択フィールドを示し、インクルード・ファイルに宣言された変数型のタイプを列挙し、それから選択する。インクルードファイルとは、モジュールファイルにその一部として組み込まれる独立したファイルであり、各モジュールに共通して用いられるもの、例えば変数型の宣言等に用いられる。13はパラメータ番号表示フィールドを示し、引数として渡されるパラメータの順番を示す。14はパラメータ追加フィールドを示し、5つ以上のパラメータを持つテスト関数に於て、画面に直接表示できないパラメータの設定を行う為のものである。15はグローバル変数入力フィールドを示し、テスト関数内部で参照されるグローバル変数の名前を設定する。16はモジュール・ファイル読み込みボタンを示し、オペレータがこれを選択すると、インターフェイス・ファイル処理部2は選択されたモジュール・ファイルを読み込み、インターフェイス・リストを作成する。17はインターフェイス・ファイル保存ボタンを示し、全ての設定をインターフェイス・ファイル5に保存する。

【0016】このようにして作成されたインターフェイス・ファイルは、テスト関数を実行するのに必要なパラメータや関数等のアドレスに関する情報を含んでおり、それに基づいて、モジュール関数自動実行処理部4は、指定されたテスト関数を実行する。モジュール定義エディタ処理部3とモジュール関数自動実行処理部4とは、連続してその処理を行うようにしてあってもよいし、夫々の処理を独立させ、オペレータがそれぞれの実行を指定することで独立して実行するようにしてあってもよい。

【0017】図3は、モジュール関数自動実行処理部により関数を実行する際の表示画面の一例を示す。

【0018】図3に於て、18はインターフェイス・ファイル選択フィールドを示し、オペレータはテストしようとする関数にかかわるインターフェイス・ファイルを選択する。19は関数名選択フィールドを示し、テストしようとする関数を選択する。20はパラメータ値入力フィールドを示し、パラメータ変数値、あるいは即値を入力する。パラメータ値入力フィールド20は、4つまでのパラメータについてはその名を羅列する形式で表示され、オペレータは羅列された名前の変数にたいする値を入力する。21はパラメータ選択フィールドを示し、5つ目以上のパラメータについての入力を実施する。22は返却値表示フィールドを示し、返却値の値を表示する。23は返却値の内容表示フィールドを示し、返却値が参照する領域の内容を表示する。24はグローバル変数選択フィールドを示し、テスト関数で参照されるグローバル変数を選択する。25は変数値表示フィールドを示し、選択されたグローバル変数の値を表示する。26はテスト関数を実行する実行ボタン、27は終了ボタンを示す。

【0019】上記各フィールドの内容としては、変数やパラメータおよびボタン類以外のものについては、モジュール定義エディタ処理部3により作成されたインターフェイス・ファイル51に登録されている値が候補として表示され、オペレータはそのうちから所望のものを選ぶことになる。

【0020】以上のようにして、オペレータはテストしようとする関数および関数のパラメータ等、実行に必要な情報を対話形式で指定することができる。このため、テスト関数を呼び出すテストプログラムをテスト関数毎に作成する手間を省くことができる。

【0021】以上、オペレータとテストシステムとのインターフェイスについて説明した。次に、モジュールテスト制御部1における各処理の手順を説明する。

【0022】図4はインターフェイス・ファイル作成処理部の処理手順を示すフローチャートである。

【0023】本実施例においては、テストの対象となるモジュールの実行ファイル（オブジェクト、ライブラリ等）がリンクされたときに、そこで作成されるマップ・ファイルが必要とする。マップファイルとは、リンクされたファイルの状態を記録したファイルであり、リンクされたファイルの名前やそのアドレス、リンク対象となる関数の名前やアドレス等が記録されている。

【0024】まず、ステップ4S1にて、予め、既定の初期状態にあるインターフェイス・ファイル（以下、DEFAULT.ITF と呼ぶ）に準備された変数用のtypedef型宣言を、図1の作業用メモリ6に読み込む。以降、ファイルの読み込み、インターフェイス・ファイルの編集等は作業用メモリ6内で行われる。DEFAULT.ITF は、図5に示すようなセクション項目を持ち、項目5-1には、C言語開発用に提供されているインクルード・ファイルに

定義されているものが含まれ、各定義語にはスタック参照の引渡しタイプ（VALUE:即値、OFFSET:オフセット値、SEGMENT:セグメント値）とサイズが指定されている。

【0025】ステップ4S2にて、指定されたモジュール・ファイルとリンク時に作成されたマップ・ファイルを読み込む。ファイルが指定されていない場合、カレント・ディレクトリにある全てのモジュール・ファイルとマップ・ファイルを読み込む。

【0026】ステップ4S3にて、モジュール・ファイルが参照している独自のインクルード・ファイルを読み込む。

【0027】ステップ4S4にて、テスト対象のモジュール・ファイル名を、図5の5-2に示す[\$\$Module_Section] 項目の項目名として登録する。

【0028】ステップ4S5にて、インクルード・ファイルで定義された変数型を示すtypedef宣言文を取り出し、ステップ4S6にて、取り出した宣言文を、図5の項目5-3に示す、[\$\$Include_Section] 項目に追加する。

【0029】ステップ4S7にて、プロトタイプ宣言文を判別するキーワードがあるか否かを判断する。このキーワードは、予め、モジュール・ファイルに明示的に記述されているものであるが、無い可能性もあるので、そのための処理としてステップ4S10以降の処理がある。判断の結果、キーワードがあれば、ステップ4S8にて、それに続くプロトタイプ宣言文を取り出す。

【0030】ステップ4S9にて、取り出したプロトタイプ宣言文から、図5の項目5-4の[\$\$Interface_Section] に示すような、インターフェイス・リスト構造体を作成する。このとき、マップ・ファイルにある同名のテスト関数を検索し、そのテスト関数のアドレスを構造体に追加する。1つのインターフェイス・リストには次のリストを示すポインタが含まれ、リレーショナルなリストが作成される。この処理をキーワードを発見できる間繰り返す。

【0031】キーワードが発見できなくなったならば、ステップ4S10にて、キーワード無しのプロトタイプ宣言文を取り出すために、読み込んだファイルの読み込みボタンを先頭に戻す。ステップ4S11にて、ファイルの終了かどうかを判断する。判断の結果、終了でなければ、ステップ4S12にて、テスト関数の中でキーワードの無いプロトタイプ宣言文を検索する。ステップ4S13にて、テスト関数が見つかったかどうかを判断し、見つかったならば、ステップ4S14にて、プロトタイプ宣言文を取り出す。ステップ4S15にて、取り出したプロトタイプ宣言文から、図5の項目5-4の[\$\$Interface_Section] に示すような、インターフェイス・リスト構造体を作成する。

【0032】判断の結果、見つからなかったならば、ス

テップ4S16にて、マップ・ファイルに記述されているグローバル変数名とアドレスを、図5の項目5-5に示す[Global_Section]に追加し、インターフェイス・ファイル処理部2における処理を終了する。

【0033】以上のようにして、インターフェイス・ファイル51には、テスト対象となるモジュール・ファイルから外部につながる関数や変数の名前及びそれらのアドレスが登録される。

【0034】図6はモジュール定義エディタ処理部3の処理手順を示すフローチャートである。これは、図2の画面に応じてユーザが操作する手順に従ったイベントの発行によって、処理が振り分けられる、イベント駆動型の処理となる。

【0035】図2のモジュール選択フィールド9の右端のスクロールボタンが押下されたならば、ステップ6S1にて、テスト対象となるモジュール・ファイル名がプルダウン形式で列挙される。ステップ6S2にて、列挙されたファイルの中から選択されたファイル名を1つ取得し、モジュール・ファイル変数として作業メモリ6に格納する。各フィールド毎に割り当てられた変数データは、保存段階で、前述のインターフェイス・ファイル作成処理部に於て作成されたインターフェイス・ファイルの各セクションの新規項目として登録される。

【0036】モジュール・ファイル読み込みボタン16が押下されると、ステップ6S14にて、選択されたモジュール・ファイルを作業メモリ6に読み込む。この時、既にインターフェイス・ファイルが存在するならば、そのインターフェイス・ファイルを読み込み、存在しなければ、ステップ6S15にて、読み込んだファイルから、前述のインターフェイス・ファイル作成処理部2を使って、インターフェイス・リストを作成する。ステップ6S16にて、インターフェイス・リストの最初のテスト関数の内容を各フィールドに割り当て、表示する。この一連の処理は、複数のモジュール・ファイルが存在するときの選択手段であり、1つしか存在しない場合は、モジュール・ファイル読み込みが自動的に実行され、最初のテスト関数についての各項目が表示される。

【0037】以下は、インターフェイス・リストの項目を変更、追加、あるいは、修正する場合の処理についてである。

【0038】関数名選択フィールド10では、右端のスクロールボタンが押下されたならば、ステップ6S3にて、テスト対象となる関数名がプルダウン形式で列挙される。ステップ6S4にて、列挙された関数の中から、選択された関数名を取得し、関数名変数として、作業メモリ6に格納する。このとき、前回とは違う別の関数が選択されたならば、前記関数についての各フィールド値をインターフェイス・リスト形式に変換して、作業メモリ6内に保存する。ステップ6S5にて、関数名をもとにして、作業メモリ6上のインターフェイス・リストか

ら関数名の合致したセクション項目を取得する。ステップ6S6にて、取得したセクションの[Interface_Section]に記述された項目を各フィールドの項目列挙変数に代入し、表示する。

【0039】変数名入力フィールド11では、ステップ6S7にて、ユーザが入力、変更した変数名を取得する。ここで入力された変数名は、インターフェイス・リストのパラメータ名に反映される。

【0040】変数型タイプ選択フィールド12では、右端のスクロールボタンが押下されたならば、ステップ6S8にて、インターフェイス・ファイル51の項目5-1、5-3に示されるtypedef宣言文がプルダウン形式で列挙される。ステップ6S9にて、選択された変数型タイプを取得し、パラメータと対にして、型タイプ変数あるいは返却値変数として作業メモリ6に保存する。

【0041】パラメータ番号表示フィールド13では、ステップ6S10にて、パラメータの引渡し順序を昇順で表示する。1~4つ目までのパラメータは、一度に表示できるが、5つ目からは、パラメータ追加フィールド14を使って、ステップ6S11にて、選択されたパラメータ順に、パラメータの設定を列挙する。ステップ6S12にて、選択された引渡し番号を取得し、上記の変数名入力フィールド、変数型タイプ選択フィールドと同様の処理を実行し、5つ目以降のパラメータの設定を行う。

【0042】グローバル変数入力フィールド15では、ステップ6S13にて、テスト関数で参照されるグローバル変数名を入力する。ステップ6S9にて、選択されたグローバル変数型タイプを取得し、変数名と変数型タイプのペアを作業メモリ6に保存する。複数のグローバル変数が参照される場合、名前とタイプが決定されたならば、右端のスクロール・ボタンを使って次の入力フィールドを表示し、新規入力する。これを、参照されるグローバル変数全てについて行う。ここ迄の処理をモジュール・ファイル内のテスト対象となる関数全てについて同様の操作手順で行う。

【0043】次に、インターフェイス・ファイル保存ボタン17では、ステップ6S16にて、作業メモリ6に保存された各テスト関数のデータをインターフェイス・リスト形式に変換して、モジュール・ファイル名の拡張子を、".ITF"に変更した名前で、インターフェイス・ファイルとして図示した外部記憶装置5に保存する。

【0044】以上の手順により、テスト関数を含むモジュールファイルのインターフェイス・ファイルを作成する。

【0045】図7はモジュール関数自動実行処理部の処理手順を示すフローチャートである。これは、ユーザの操作手順に従ったイベントの発行によって、処理が振り分けられる、イベント駆動型の処理となる。

【0046】図3のインターフェイス・ファイル選択フ

ィールド18では、右端のスクロール・ボタンが押下されたならば、ステップ7S1によって、上記モジュール定義エディタによって作成されたインターフェイス・ファイルをプルダウン形式で列挙する。ステップ7S2にて、ユーザによって選択されたインターフェイス・ファイル名を取得する。ステップ7S3にて、インターフェイス・ファイルを作業メモリ6に読み込み、[`$$$Interface_Section`]に登録されたテスト関数の名前を、関数名選択フィールド19の列挙項目とする。

【0047】右端のスクロール・ボタンが押下されたならば、ステップ7S4にて、ステップ7S3で設定されたテスト関数名をプルダウン形式で列挙する。ステップ7S5にて、列挙された関数名から選択されたテスト関数名を取得する。ステップ7S6にて、テスト関数に応じたインターフェイス・リストから[`$$$Interface_Section`]項目を取り出し、その中の1~4つ迄の入力パラメータ名を表示し、5つ目以降のパラメータ名をパラメータ選択フィールド21の列挙項目に加える。

【0048】ステップ7S7にて、パラメータの値を入力値フィールド20に入力する。入力された値は、インターフェイス・リストのパラメータ型タイプによって、アドレス参照か、即値か決定される。パラメータ選択フィールド21とそれに続くパラメータ値入力フィールド20には、ステップ7S8にて、5つ目以降のパラメータ名を列挙し、ステップ7S9にて、パラメータ値を順次入力する。この結果、5つ目以降のパラメータについての入力操作ができる。

【0049】グローバル変数選択フィールド24では、ステップ7S12にて、インターフェイス・ファイルの項目5-5に登録されたグローバル変数名が列挙される。変数値表示フィールド25では、ステップ7S13にて選択された変数名を取得し、ステップ7S14にて変更される前の値を設定する。このフィールドは関数実行前は値の入力を受け付け、実行後変更された変数の値を表示する。ここまでの処理でテスト関数のパラメータ入力が終了する。

【0050】次に、実行ボタン26が押下され、ステップ7S16にて、テスト関数が実行される。詳細は、図8に示す。

【0051】ステップ7S16-1にて、設定されたパラメータ値及びグローバル変数値を取得する。ステップ7S16-2にて、パラメータ変数用の作業領域を作業メモリ6に確保する。ステップ7S16-3にて、テスト関数名からインターフェイス・ファイルに記述されたセクション項目の内容を作業メモリ6に取り出す。このセクション項目のパラメータ・タイプ、関数アドレスに従って、テスト関数が実行される。

【0052】ステップ7S16-4にて、テスト関数内で参照されるグローバル変数があるかを判断する。判断の結果、参照があれば、ステップ7S16-5にて、上

記処理で入力された変数値を設定する。

【0053】ステップ7S16-6にて、パラメータの引渡しが終わったかを判断する。判断の結果、終了でなければ、ステップ7S16-7にて、読み込んだインターフェイス・ファイルから、テスト関数に引き渡される全てのパラメータ・タイプを取得する。ステップ7S16-8にて、パラメータがセグメント間の参照かを判断する。判断の結果、セグメント参照であるならば、ステップ7S16-9にて、作業メモリ6に割り当てられたDATAセグメントの値をスタックにPUSHする。セグメント参照でなければ、ステップ7S16-10にて、アドレス参照かを判断する。判断の結果、アドレス参照ならば、ステップ7S16-11にて、パラメータのオフセット値をスタックにPUSHする。アドレス参照でなければ、ステップ7S16-12にて、パラメータの即値をスタックにPUSHする。

【0054】以上の処理を、パラメータの引渡しが続く間行い、ステップ7S16-6の判断によって、引渡しが終わったならば、ステップ7S16-13にて、インターフェイス・ファイルの関数名に対応したアドレス値から、テスト関数のCODEセグメントと、テスト関数の呼び出しアドレスを取得し、ステップ7S16-14にて、テスト関数を実行する。処理の結果は、ステップ7S16-15にて、返却値、グローバル変数の表示用のフィールド項目に設定する。

【0055】ステップ7S16の詳細は以上のようなものである。

【0056】ステップ7S10にては、返却値の値を表示する。ステップ7S11にて、返却値がアドレス参照値であれば、参照される領域の内容を表示する。また、グローバル変数の変更結果については、入力手順と同様に、ステップ7S12、13にて、グローバル変数を選択し、ステップ7S15にて、選択されたグローバル変数の値を表示する。以上の処理によって、テスト対象となる関数単位でのテストが可能になる。

【0057】このように、本実施例のテストシステムでは、モジュール・ファイルに対応したインターフェイス・ファイルを準備することで、いかなるモジュールに対しても統一的な操作環境を提供し、モジュール関数単体の実行テストを可能にする。オペレータはモジュールの内容にかかわらず統一された形式の手順で、統一された画面表示を見ながらテスト対象となるモジュールの検証を行うことができる。そのため、モジュールごとにテストプログラムの作成といった作業が不要となり、効率的にモジュールテストを行うことができる。

【0058】また、オペレータはメニュー形式の画面を参照しつつ対話形式でテストを遂行することができるため、パラメータの値変更やテスト対象の関数の変更が容易にでき、効率的にテストを行うことができる。

【0059】また、本テストシステムは、プログラムの

開発環境と同一のコンピュータシステム内に備えることができる。そのため、モジュール開発後時間をおかずにそのテストを行うことができる。

【0060】

【他の実施例】本実施例では、モジュール関数自動実行処理部4の中に関数呼び出しの機構を設けているが、この機構をコマンド・インタプリタに切り替えることで、自作したコマンド・ユーティリティ、あるいは、ソース・ファイルの無い、市販のパラメータ入力を持ったコマンド・ユーティリティに対して単体テストの環境を提供することができる。

【0061】また、本実施例はC言語で作成されたプログラムに関して説明したが、他の言語で記述された場合であってもサブルーチン単位でモジュール化されることは通常行われることであり、そのような場合にも各モジュールファイル間をつなぐための変数やサブルーチンなどの名前やアドレスに関する情報、すなわちテスト態様となるプログラムモジュールとそれを用いるプログラムとのインターフェイス情報を用い、パラメータとして適当な値をセットさせ、テスト対象プログラムを呼び出せば、本実施例と同様にテストシステムを構成することができることは容易に理解できる。

【0062】尚、本発明は、複数の機器から構成されるシステムに適用しても、1つの機器から成る装置に適用しても良い。また、本発明はシステム或は装置にプログラムを供給することによって達成される場合にも適用できることは言うまでもない。

【0063】

【発明の効果】以上説明したように、本発明に係るモジュールテスト方法およびそれを用いた装置とシステムによれば、プログラムの開発において統一的なモジュール

単体テストが可能になり、開発効率を向上させるという効果を奏する。

【0064】

【図面の簡単な説明】

【図1】本発明の実施例を示すテストシステムのブロック図である。

【図2】ダイアログ型モジュール定義エディタの表示画面の概要図である。

【図3】モジュール関数自動実行エディタによる表示画面の概要図である。

【図4】インターフェイス・ファイル作成処理部のフローチャートである。

【図5】インターフェイス・ファイルの書式である。

【図6】モジュール定義エディタ処理部のフローチャートである。

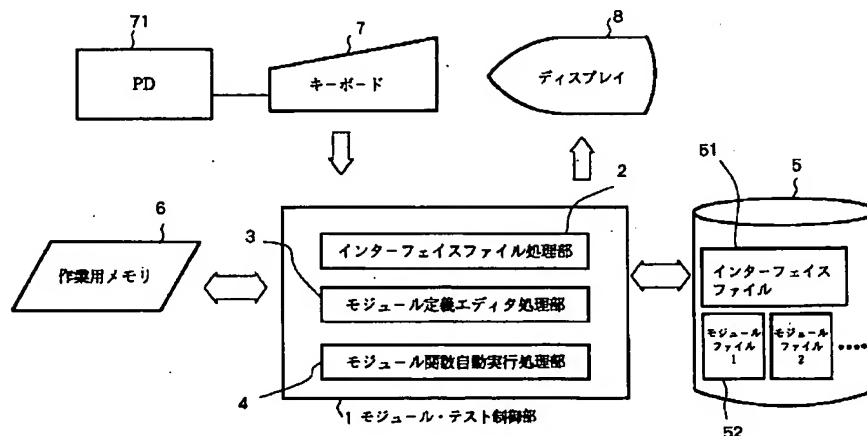
【図7】モジュール関数自動実行処理部のフローチャートである。

【図8】モジュール関数自動実行処理部のフローチャートである。

【符号の説明】

- 1 モジュール単体テスト制御部
- 2 インターフェイス・ファイル作成処理部
- 3 ダイアログ型モジュール定義エディタ処理部
- 4 モジュール関数自動実行処理部
- 5 インターフェイス・ファイル（外部記憶装置）
- 6 作業用メモリ
- 7 キーボード
- 8 ディスプレイ
- 51 インターフェイスファイル
- 52 モジュールファイル（外部記憶装置）
- 71 ポインティング・デバイス

【図1】



【図2】

【図5】

インターフェイス・ファイル書式図

5-1
DEFAULT, ITPC設定
仕様

5-2

5-3
外部-呼び出し定義
仕様 typedef宣言文

5-4
外部-呼び出し定義
仕様 typedef宣言文

5-5
グローバル変数の定義

```

[Default_Section]
INT      VAL : 2
WORD     VAL : 2
BYTE     VAL : 2
DWORD    VAL : 4
HANDLE   VAL : 2
PTR      OFF : 2
FPTR     SEG : 2, OFF : 2
...

[$Module_Section]

[$Include_Section]
HANDLE   DWORD
PTR      PTR
LPSTR    FPTR
...

[$Interface_Section]
INT Print : Address (
PTR      pString
INT      row :
INT      col :
PTR      next_section ;
)

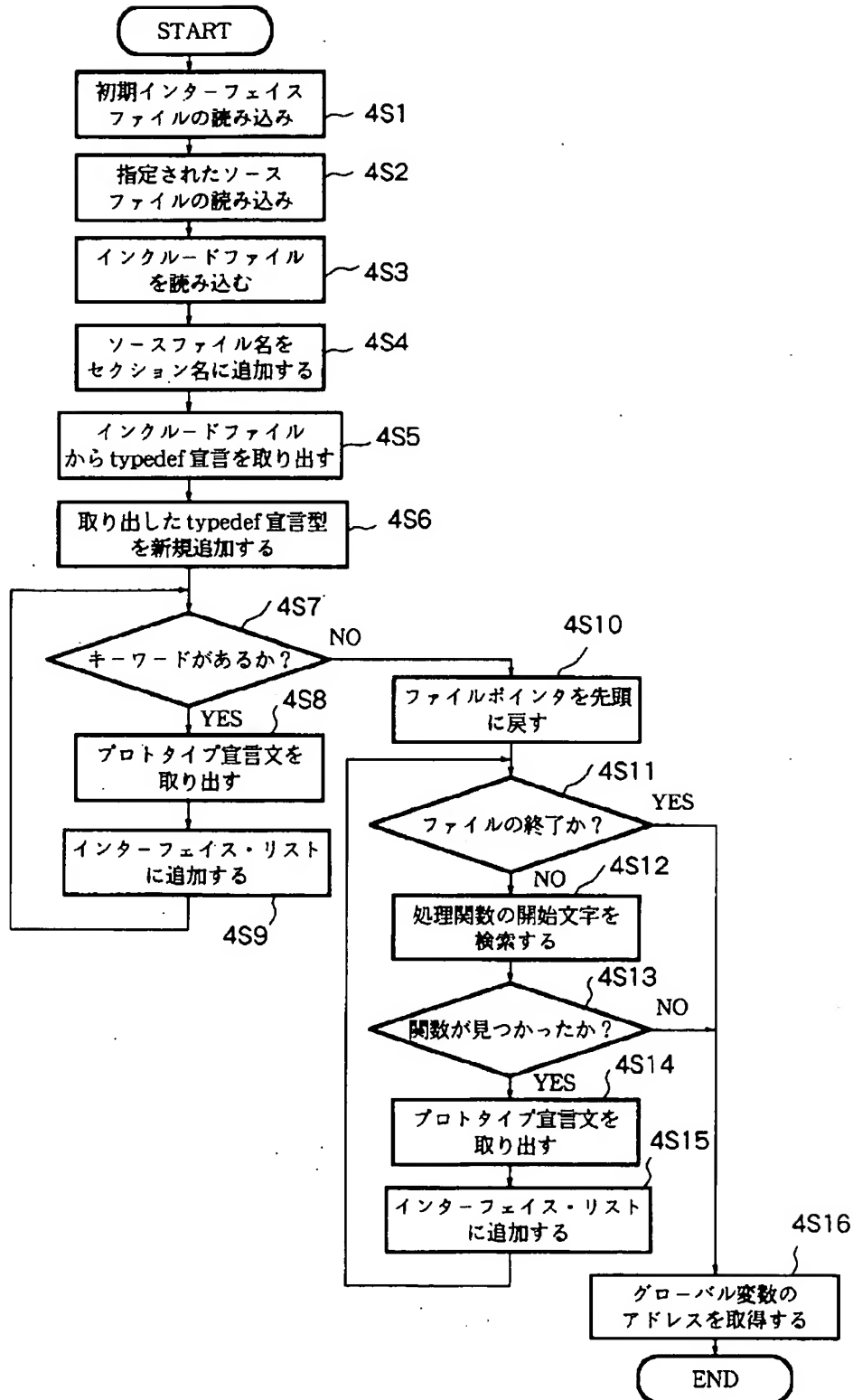
INT Cursor : Address (
INT      row :
INT      col :
PTR      next_section ;
)

HANDLE GetHandle : Address (
WORD size :
INT      flag :
PTR      next_section ;
)
:

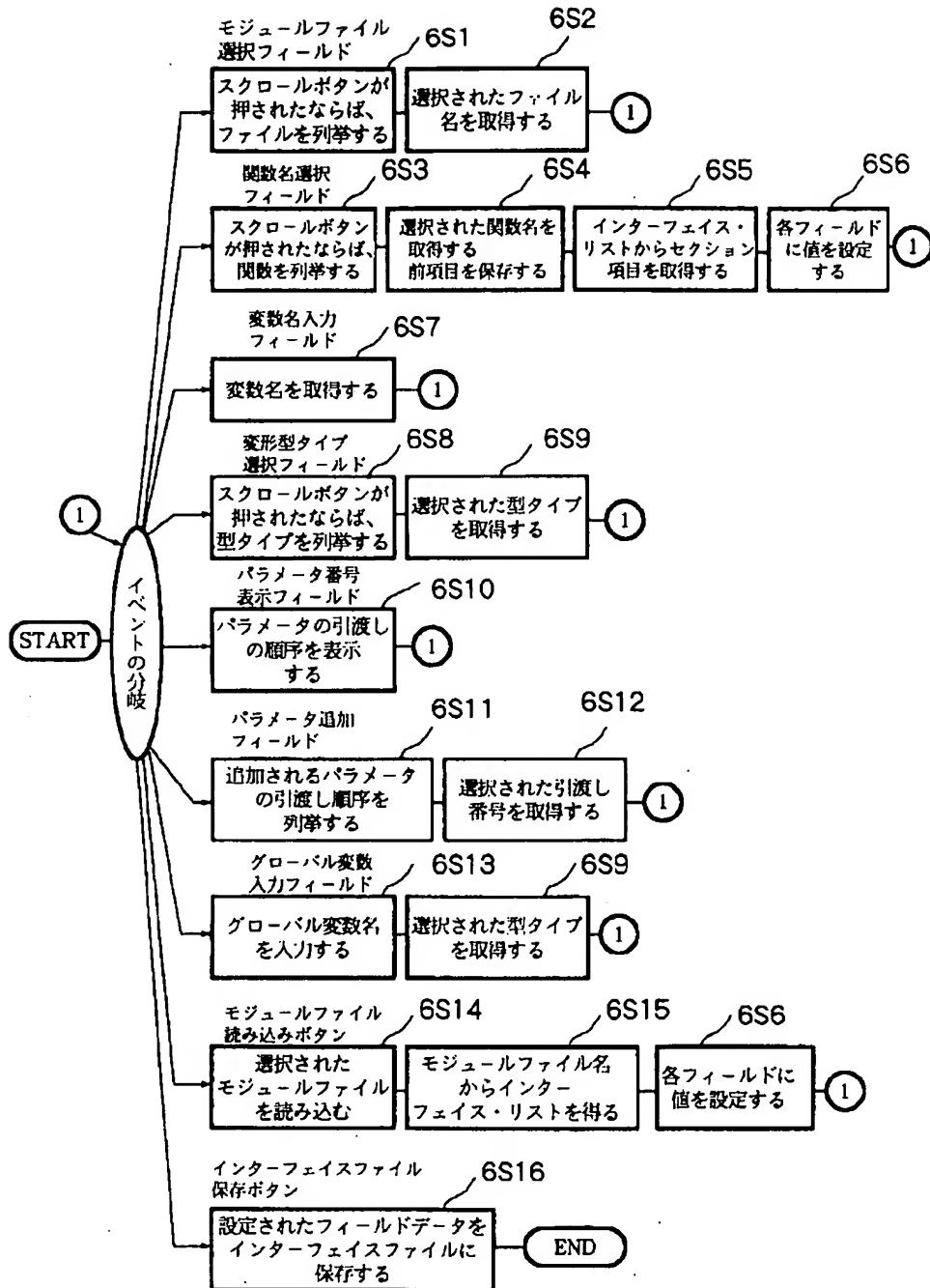
[$Global_Section]
INT      count : Address
WORD     msg : Address
...
  
```

【図3】

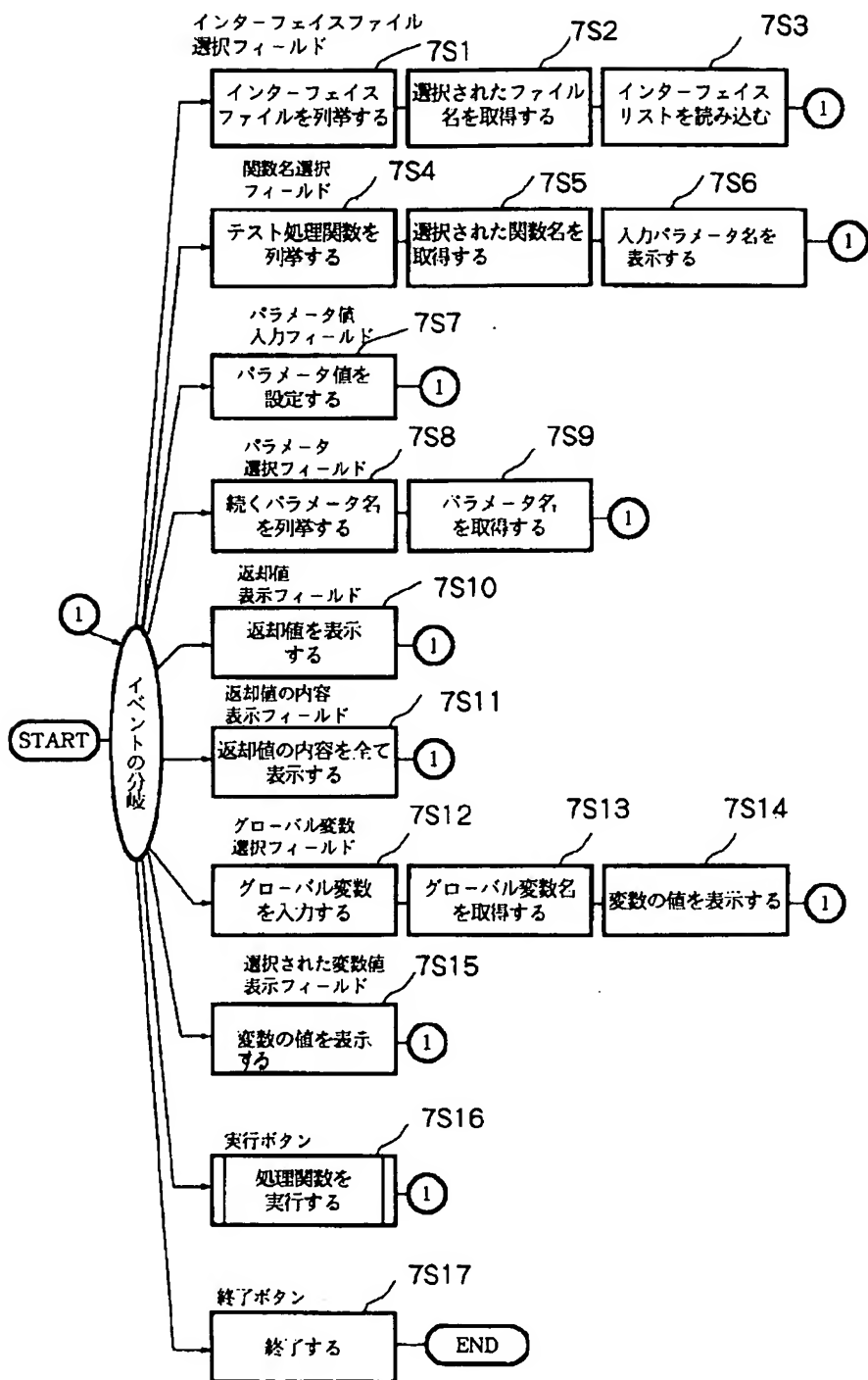
【図4】



【図6】



【圖 7】



【図8】

